EEL 6764 001: Graduate Computer Architecture Spring 2025
Instructor: Dr. Srinivas Katkoori

Homework 1 Solutions

1. Solution:
   a. Given, For "A"
      Parallelizable = 50 = 50% = 0.5
      It's a 22-core processor,
      So, Speedup = $\frac{1}{(0.5+\frac{0.5}{22})}$ = 1.91
   b. Given, For "D"
      Parallelizable = 90 = 90% = 0.9
      It's a 22-core processor,
      So, Speedup = $\frac{1}{(0.1+\frac{0.9}{22})}$ = 7.1
   c. A requires 41% of resources,
      So, 41% * 22 = 9
      Speedup of A on 9 cores = $\frac{1}{(0.5+\frac{0.5}{9})}$ = 1.8
      Overall speedup if 9 cores have 1.8 and others none is
      Speedup = $\frac{1}{(0.6+\frac{0.4}{1.8})}$ = 1.22
   d. A requires 41% of resources,
      So, 41% * 22 = 9
      Speedup of A on 9 cores = $\frac{1}{(0.5+\frac{0.5}{9})}$ = 1.8
      B requires 27% of resources,
      So, 27% * 22 = 5.9
      Speedup of B on 5.9 cores = $\frac{1}{(0.2+\frac{0.8}{5.9})}$ = 3
      C requires 18% of resources,
      So, 18% * 22 = 3.96
      Speedup of C on 9 cores = $\frac{1}{(0.4+\frac{0.6}{3.96})}$ = 1.82
      D requires 14% of resources,
      So, 14% * 22 = 3.08
      Speedup of D on 9 cores = $\frac{1}{(0.1+\frac{0.9}{3.08})}$ = 2.5

2. True or False
    a. **False**: The performance of the system is limited by the slowest component not by the fastest. If some components become 7 times slower, then the overall performance is affected.
    b. True: Amdahl's is still applicable in field of computer science and in parallel computing. It helps in understanding the limits of performance improvement when attempting a program to run faster by parallelizing it.
    c. False: CPI rating alone is not sufficient to measure the performance of the system, we also need clock speed, instruction mix etc.
    d. True: Adding multiple cores to the chip becomes expensive, instead we can use software to implement the parallelization.

3. Fabrication Cost
    a. Phoenix:

$$Dies\ per\ wafer = \frac{\pi \times (\frac{Wafer\ diameter}{2})^2}{Die\ Area} - \frac{\pi \times \text{Wafer Diameter}}{\sqrt{2 \times Die\ Area}}$$

$$= \frac{\pi \times (\frac{45}{2})^2}{2} - \frac{\pi \times 45}{\sqrt{2 \times 2}}$$

$$= 795 - 70.7 = 724.5 = 724$$

$$Yield = \frac{1}{(1 + Defects\ per\ unit\ area\ \times Die\ Area)^N}$$

$$= \frac{1}{(1+0.04 \times 2)^{14}} = 0.340$$

Profit = 724 * 0.34 * 30 = $7384.80

    b. Red Dragon:

$$Dies\ per\ wafer = \frac{\pi \times (\frac{Wafer\ diameter}{2})^2}{Die\ Area} - \frac{\pi \times \text{Wafer Diameter}}{\sqrt{2 \times Die\ Area}}$$

$$= \frac{\pi \times (\frac{45}{2})^2}{2} - \frac{\pi \times 45}{\sqrt{2 \times 1.2}}$$

$$= 1325 - 91.25 = 1234$$

$$Yield = \frac{1}{(1 + Defects\ per\ unit\ area\ \times Die\ Area)^N}$$

$$= \frac{1}{(1+0.04 \times 1.2)^{14}} = 0.519$$

Profit = 1234 * 0.519 * 15 = $9601.71

c. For Phoenix Chips = $\frac{25000}{724}$ = 34.5

For Red dragon chips = $\frac{50000}{1234}$ = 40.5

So, if the facility can fabricate 70 wafers per month, then it fabricates 30 phoenix chips and 40 red dragon chips.

4. Solution

a. No leakage at idle time, so $P_s = 0$

Power = $P_d + P_s$
= P_d
= 1 W * 4 = 4 W

Energy = Power * Time
E_s = 0 * T = 0
E_d = 4T

When processor runs for full time energy = 4T

Given the processor runs for ½ of the time.

= 4 * T/2 = 2T

Therefore, $\frac{Energy\ when\ operating\ for\ \frac{1}{2}\ time}{Energy\ when\ operating\ for\ full\ time}$ = $\frac{2T}{4T}$ = ½

Energy when operates for ½ time is ½ of energy when runs for full time.

Power:

There will be no change in power because the device runs for half of the time and energy is consumed half. So, Power = Energy / Time. Therefore power becomes constant.

b. Energy = K * C * V^2

Power = $K * C * V^2 * f$

Given frequency and voltage are reduced to ¼ of the time.

Energy old = $K * C * V^2$

Energy new = $K * C * (V/4)^2$

Dynamic Energy = Energy New / Energy old

$= K * C * (V/4)^2 / K * C * V^2$

$= 1 / 16$

$= 0.0625$

Power = $K * C * V^2 * f$

Power old = Power = $K * C * V^2 * f$

Power new = Power = $K * C * (V/4)^2 * f / 4$

Dynamic Power = Power new / Power old

$= (K * C * (V/4)^2 * f / 4) / K * C * V^2 * f$

$= (1/ 16) * (1/ 4)$

$= 0.0156$

c. Voltage can go below up to 70% = 0.7 V

Energy old = $K * C * V^2$

Energy new = $K * C * (0.7V)^2$

Dynamic Energy = Energy New / Energy old

$= K * C * (0.7V)^2 / K * C * V^2$

$= 0.49$

Frequency can go upto ¼ th of time = f / 4

Power = $K * C * V^2 * f$

Power old = Power = $K * C * V^2 * f$

Power new = Power = $K * C * (0.7\ V)^2 * f / 4$

Dynamic Power = Power new / Power old

$= (K * C * (0.7V)^2 * f / 4) / K * C * V^2 * f$

$= (0.49) * (1/ 4)$

$= 0.1225$

d. For processor with 4 cores, Energy required be "E "

Since only 1 core is used,

So, Energy = E / 4

For first 25%

Energy (E_1) = (25 / 100) * (E /4)

For remaining 75%

Energy (E_2) = (75 / 100) * (E /4) * (20 /100) (Because, 20 % of the energy is used during 75%)

Total Energy (E) = Energy (E_1) + Energy (E_2)

$= ((25 / 100) * (E /4)) + ((75 / 100) * (E /4) * (20 /100))$

$= 0.1\ E$

5. Effective CPI = ∑Instruction category frequency * Clock cycles for category

   Effective CPI = (0.395)×(1.0))+((0.28)×(3.5))+((0.115)×(2.8) +((0.19)×((0.5) × (4.0)+(0.5)×(2.0)))+(0.015×2.4)+(0.005×3.0)
   = 0.395+0.98+0.322+0.57+0.036+0.015 = 2.318

6. For this example, let's examine the x86 architecture, which is one of the most well-known CISC (Complex Instruction Set Computer) architectures.
   a. The x86 architecture is considered a CISC architecture primarily due to its extensive instruction set. It offers a wide variety of complex instructions that can perform multiple operations in a single instruction. These instructions often involve memory access, arithmetic operations, and control flow.
   b. MOV (Move): MOV instruction can perform a wide range of tasks, including data transfers between registers and memory, type conversions, and more.
      LEA (Load Effective Address): This instruction calculates and loads the effective address of a memory operand, allowing for complex address calculations.
      STRING instructions: Instructions like MOVS, LODS, STOS, etc., are used for string manipulation and can perform multiple memory operations in a single instruction.
      XLAT (Translate): This instruction performs a table lookup and is used for character translation.

   c. Register addressing: Direct access to registers (e.g., MOV AX, BX).
      Immediate addressing: Constants or immediate values are used as operands (e.g., MOV AX, 42).
      Memory addressing modes:
      Direct memory addressing (e.g., MOV AX, [BX]).
      Indexed addressing (e.g., MOV AX, [BX+SI]).
      Base addressing (e.g., MOV AX, [BX+42]).
      Scaled indexing (e.g., MOV AX, [BX+SI*2]).
      Based indexed addressing (e.g., MOV AX, [BX+SI]+42).
      Register indirect addressing: Using a register to access memory (e.g., MOV AX, [BX]).
      Base pointer addressing: Using a base pointer register (e.g., MOV AX, [BP+SI]).
      Stack addressing: Using the stack pointer register (e.g., PUSH AX, POP BX).
      Immediate addressing with displacement (e.g., MOV AX, 42[SI]).

7. Given
   IF, 2 ns; ID, 3.5 ns; EX, 1 ns; MEM, 4 ns; and WB, 2.5 ns.
   a. Clock time for pipelined machine = max-time + delay
      Memory has max time
      Clock time = 4 + 0.1 = 4.1
   b. Given that there is stall for every 4 instructions, so for every 5 cycles 4 instructions are executed. One cycle will be wasted for stall.
      CPI = Clock Cycles / Instruction Count
      = 5 /4 = 1.25
   c. For single processor
      Clock Period = 13
      Assume CPI = 1
      For Pipelined processor Clock Period = 4.1
      CPI = 1.25
      Speedup = $\frac{Execution\ time\ for\ non-pipelined\ architecture}{Execution\ time\ for\ pipelined\ architecture}$
      Execution Time = Instruction * CPI * Clock Period
      Speedup = $\frac{I*1*13}{I*1.25*4.1}$
      = 13 / 5.125 = 2.5
   d. For single processor
      Clock Period = 13
      Assume CPI = 1
      For Pipelined processor with infinite number of stages
      Clock Period = 0.1
      CPI = 1
      Speedup = $\frac{Execution\ time\ for\ non-pipelined\ architecture}{Execution\ time\ for\ pipelined\ architecture}$
      Execution Time = Instruction * CPI * Clock Period
      Speedup = $\frac{I*1*13}{I*1*0.1}$
      = 130

8. The four types of Flynn's taxonomy are:
   1. SISD (Single Instruction Single Data)
      This design is an example of the classic von Neumann architecture, where a single processor executes one instruction at a time on a single piece of data.
   2. SIMD (Single Instruction Multiple Data)
      Here a single instruction is used to operate on multiple data elements simultaneously. Parallelism is achieved through data-level parallelism, where multiple processing units execute the same instruction on different data elements.
   3. MISD (Multiple Instruction Single Data)
      MISD architectures are less common and involve multiple instructions acting on the

same data stream. While they do exist theoretically, they are not widely used in practice because it's challenging to design scenarios where multiple different instructions operate on the same data effectively.

4. MIMD (Multiple Instruction, Multiple Data)
   MIMD architectures involve multiple processors or cores, each capable of executing | its own set of instructions on its own data.

For
SISD: No parallelism exploited.
SIMD: Data-level parallelism exploited.
MISD: Rarely used, limited practical applications.
MIMD: Exploits both instruction-level and data-level parallelism.